Attorney Docket No.: 021751-000710US

PATENT APPLICATION

MODEL REFERENCING METHOD AND APPARATUS

Inventor(s): Robert Jensen, a citizen of the US
residing at:
1805 Delaware Street
Berkeley, CA 94703

Andrew Witkin, a citizen of the US
residing at:
5552 Golden Gate Avenue
Oakland, CA 94618

Assignee: Pixar
1200 Park Avenue
Emeryville, CA, 94608

Entity: Large

TOWNSEND and TOWNSEND and CREW LLP
Two Embarcadero Center, Eighth Floor
San Francisco, California 94111-3834
Tel: 650-326-2400

MODEL REFERENCING METHOD AND APPARATUS

CROSS-REFERENCES TO RELATED APPLICATIONS

[0001] The present application incorporates by reference for all purposes and claims priority to Provisional Application No. 60/470948, filed May 14, 2003.

BACKGROUND OF THE INVENTION

[0002] The present invention relates to computer animation. More specifically, the present invention relates to management of animation objects.

[0003] Throughout the years, movie makers have often tried to tell stories involving make-believe creatures, far away places, and fantastic things. To do so, they have often relied on animation techniques to bring the make-believe to "life." Two of the major paths in animation have traditionally included, drawing-based animation techniques and stop motion animation techniques.

[0004] Drawing-based animation techniques were refined in the twentieth century, by movie makers such as Walt Disney and used in movies such as "Snow White and the Seven Dwarves" and "Fantasia" (1940). This animation technique typically required artists to hand-draw (or paint) animated images onto a transparent media or cels. After painting, each cel would then be captured or recorded onto film as one or more frames in a movie.

[0005] Stop motion-based animation techniques typically required the construction of miniature sets, props, and characters. The filmmakers would construct the sets, add props, and position the miniature characters in a pose. After the animator was happy with how everything was arranged, one or more frames of film would be taken of that specific arrangement. Stop motion animation techniques were developed by movie makers such as Willis O'Brien for movies such as "King Kong" (1932). Subsequently, these techniques were refined by animators such as Ray Harryhausen for movies including "The Mighty Joe Young" (1948) and Clash Of The Titans (1981).

[0006] With the wide-spread availability of computers in the later part of the twentieth century, animators began to rely upon computers to assist in the animation process. This included using computers to facilitate drawing-based animation, for example, by painting images, by generating in-between images ("tweening"), and the like. This also included

using computers to augment stop motion animation techniques. For example, physical models could be represented by virtual models in computer memory, and manipulated.

[0007] One of the pioneering companies in the computer aided animation (CAA) industry was Pixar, dba Pixar Animation Studios. Over the years, Pixar developed and offered both computing platforms specially designed for CAA, and rendering software now known as RenderMan®. RenderMan® renders images based upon geometric scene descriptors including references to object models. Between different scenes, the objects are typically repositioned, or animated by an animator. To specify the animation of the object, the animator uses one or more object rigs to manipulate the object in time.

[0008] One method for creating an object is via creating an object from scratch. For example, the user specifies an object from the ground-up. Such a technique is of course inefficient, as often objects have portions that could be reused by other users for other objects or the user could use portions of other users' objects.

[0009] One method for creating an object, not necessarily in the prior art, is with the use of templates as starting points for customization. For example, to create an object, a user may retrieve a template for the object, customize it, and then save the customized object. As another example, a user may use templates of more than one object as a starting point to create a larger object, and then customize the objects. In such cases, after customization, the objects that are specified by the templates are stored as part of the created object.

[0010] Disadvantages to these approaches include that subsequent changes to the original model template, after the object template was used, are not propagated to the created object. Instead, any changes to the object template would have to be manually propagated to the objects that used the template. Additionally, disadvantages include that users could customize an object so significantly that it would effectively be useless to other users.

[0011] The inventors of the present invention have determined that improved methods for crating and updating objects are needed without the drawbacks illustrated above.

BRIEF SUMMARY OF THE INVENTION

[0012] The present invention relates to computer animation. More specifically, the present invention relates to referencing of models in a scene.

[0013] Model referencing allows building of computer graphics models that incorporate re-usable components, with the ability to customize the components, and to automatically receive updates as the original components are modified.

[0014] Character models are complex and unique, but have many features in common. However, the inventors have determined that it was very inefficient to duplicate and independently maintain these common features. Model referencing allows the common elements to be shared, but still allows the unique features of each character to be captured.

[0015] The effort to build shared components, such as a hand, face, or body object ("rig"), is amortized over many characters. Fixes and updates to the shared rig propagate to all characters automatically. It is much easier to maintain a standardized animator interface, facilitating animators' training and working efficiency. The quality and complexity of background characters, which must be built under stringent budget constraints, is far higher than it was when each character was made from scratch or from a template.

[0016] Customizations to shared model components are stored as overrides that describe the changes to those components from their original state. Each time a new version of the shared component is brought in, the stored overrides are re-applied to the component, so that the model both receives the updates and retains its customizations. The author of a shared component selects which internal parts will be exposed for customization when models reference the component, providing an API for customization, and making the re-application of overrides more robust.

[0017] There are several aspects to working with shared model components: creating or authoring a shared component, using or referencing a shared component, and re-merging and applying parameter overrides.

[0018] Authoring a shared component: A shared component consists of a conventionally constructed model or model fragment, augmented with additional information that determines how that model will appear when referenced by other models. Objects may be shared as *public*, indicating that the object's attributes will be accessible for editing by the author of the model that references the shared component. By default, objects are private, meaning they will be invisible and their attributes cannot be edited. In addition, objects which have been marked public may have a *path marker* set. The path marker is used to control the way objects are identified for the purpose of applying overrides.

[0019] Referencing a shared component: The author of a model that references a shared component (which may itself by used recursively as a shared component of other models) first declares a new instance of the shared component, specifying where in the model's hierarchic structure the new instance will be attached. The file containing the definition of the component is located and opened for reading only, and the model it contains is copied into the local model. The author continues working on the model. Referenced objects that had been marked public are visible and can be edited in the usual way. These edits are stored as overrides that are saved as part of the local model, and can be re-applied when a new version of the referenced model is obtained. Referenced objects that had been marked private are hidden and cannot be edited.

[0020] Re-merging and applying overrides. When a new version of a shared component is released, the author of a model that references that component can merge in the new version. The merge process proceeds as follows: the file containing the shared component is opened read only. In some embodiments, all the new objects in the shared component are copied into the current local model, alongside the old foreign objects that had been produced by the previous merge operation. In other embodiments, not all old foreign objects are saved.

[0021] Each foreign public object is located using the specification of its path, in one embodiment. If the object is not located, the user is warned and given the opportunity to fix the local model so that the object can be found. Overrides are then applied to the object. In some cases, the new and old copies of the object are compared as an aid to applying the override. When all the overrides have been applied, local objects that refer to foreign objects are modified to refer to the new ones, and finally the old foreign objects can be deleted.

[0022] According to one aspect of the invention, a method for a computer system is described. One technique includes opening a first file describing a first object in an object environment, determining a reference for a second object, wherein the second object includes a plurality of attributes. Another technique also includes receiving a second file describing the second object from a storage system; in response to the reference, and opening the second file describing the second object in the object environment. The process may also include determining a modified value for an attribute from the plurality of attributes for the second object, and including in the first file the reference for the second object and the modified value for the attribute. In various embodiments, the second object is not stored in the first file.

[0023] According to yet another aspect of the invention, a computer system including an object environment is disclosed. One system includes a storage system configured to store a first file describing a first object and a second file describing a second object, wherein the storage system is also configured to provide the first file in response to a first reference and configured to provide the second file in response to a second reference. The system may also include a processor coupled to the storage system, wherein the processor is configured to open the first file, wherein the processor is configured to determine the second reference in response to the first file, wherein the processor is configured to determined a value of an attribute from the second object in response to the first file, wherein the processor is configured to provide the second reference to the storage system, wherein the processor is configured to receive the second file from the storage system, wherein the processor is configured to open the second file, and wherein the processor is configured to override a default value of the attribute from the second object with the value. In various embodiments, a model of the second object is not stored in the first file.

[0024] According to one aspect of the invention, a computer program product for a computer system including a processor coupled to a server is taught. Various computer code includes code that directs the processor to allow a user to create a first object in an object environment, code that directs the processor to determine a reference to a second object in the server, wherein the second object includes a plurality of attributes, and code that directs the processor to create an instance of the second object in the object environment. Various embodiments also include code that directs the processor to determine a modified value for an attribute from the plurality of attributes for the second object, and code that directs the processor to override a default value for the attribute with the modified value. In some embodiments, the attribute of second object stored in the server is not modified. The computer codes typically reside on a tangible media such as a computer hard drive, network drive, optical media (CD-ROM, DVD), and the like

BRIEF DESCRIPTION OF THE DRAWINGS

[0025] In order to more fully understand the present invention, reference is made to the accompanying drawings. Understanding that these drawings are not to be considered limitations in the scope of the invention, the presently described embodiments and the presently understood best mode of the invention are described with additional detail through use of the accompanying drawings in which:

[0026] Fig. 1 illustrates a block diagram of a rendering system according to one embodiment of the present invention;

[0027] Fig. 2 illustrates a block diagram of an embodiment of the present invention;

[0028] Fig. 3 illustrates a block diagram of a flow chart according to embodiments of the present invention;

[0029] Figs. 4-B illustrate another block diagram of a flow chart according to an embodiment of the present invention; and

[0030] Figs. 5A-B illustrate a block diagram of a flow process according to an embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[0031] Fig. 1 is a block diagram of typical computer rendering system 100 according to an embodiment of the present invention.

[0032] In the present embodiment, computer system 100 typically includes a monitor 110, computer 120, a keyboard 130, a user input device 140, a network interface 150, and the like.

[0033] In the present embodiment, user input device 140 is typically embodied as a computer mouse, a trackball, a track pad, wireless remote, and the like. User input device 140 typically allows a user to select objects, icons, text and the like that appear on the monitor 110.

[0034] Embodiments of network interface 150 typically include an Ethernet card, a modem (telephone, satellite, cable, ISDN), (asynchronous) digital subscriber line (DSL) unit, and the like. Network interface 150 are typically coupled to a computer network as shown. In other embodiments, network interface 150 may be physically integrated on the motherboard of computer 120, may be a software program, such as soft DSL, or the like.

[0035] Computer 120 typically includes familiar computer components such as a processor 160, and memory storage devices, such as a random access memory (RAM) 170, disk drives 180, and system bus 190 interconnecting the above components.

[0036] In one embodiment, computer 120 is a PC compatible computer having multiple microprocessors such as Xeon™ microprocessor from Intel Corporation. Further, in the present embodiment, computer 120 typically includes a UNIX-based operating system.

[0037] RAM 170 and disk drive 180 are examples of tangible media for storage of data, audio / video files, computer programs, applet interpreters or compilers, virtual machines, scene descriptor files, object data files, shader descriptors, a rendering engine, output image files, texture maps, displacement maps, object creation environments, animation environments, asset management systems, databases and database management systems, and the like. Other types of tangible media include floppy disks, removable hard disks, optical storage media such as CD-ROMS and bar codes, semiconductor memories such as flash memories, read-only-memories (ROMS), battery-backed volatile memories, networked storage devices, and the like.

[0038] In the present embodiment, computer system 100 may also include software that enables communications over a network such as the HTTP, TCP/IP, RTP/RTSP protocols, and the like. In alternative embodiments of the present invention, other communications software and transfer protocols may also be used, for example IPX, UDP or the like.

[0039] Fig. 1 is representative of computer rendering systems capable of embodying the present invention. It will be readily apparent to one of ordinary skill in the art that many other hardware and software configurations are suitable for use with the present invention. For example, the use of other micro processors are contemplated, such as Pentium™ or Itanium™ microprocessors; Opteron™ or AthlonXP™ microprocessors from Advanced Micro Devices, Inc; PowerPC G3™, G4™ microprocessors from Motorola, Inc.; and the like. Further, other types of operating systems are contemplated, such as Windows® operating system such as WindowsXP®, WindowsNT®, or the like from Microsoft Corporation, Solaris from Sun Microsystems, LINUX, UNIX, MAC OS from Apple Computer Corporation, and the like.

[0040] Fig. 2 illustrates a block diagram of an embodiment of the present invention. Specifically, Fig. 2 illustrates an animation environment 200, an object creation environment 210, and a storage system 220.

[0041] In the present embodiment, object creation environment 210 is an environment that allows users (modellers) to specify object articulation models, including armatures and rigs. Within this environment, users can create models (manually, procedurally, etc.) of objects and specify how the objects articulate with respect to animation variables (Avars). In one specific embodiment, object creation environment 210 is a Pixar proprietary object creation environment known as "Gepetto." In other embodiements, other types of object creation environments can be used.

[0042] In the present embodiment, the object models that are created with object creation environment 210 may be used in animation environment 200. For example, as illustrated in Fig. 2, a character "Bob" references a generic character rig 240. Additionally, object models may be referenced by other object models. For example, as illustrated in Fig. 2, a heirarchial object model structure 230 is supported. Structure 230 may represent a bipedal character, for example, that includes a body model (generic character rig) 240, arm model (generic arm rig) 250, leg model (generic leg rig) 260, upper arm model (generic upper arm fig) 270, lower arm model (generic lower arm rig) 280, hand model (generic hand rig) 290, and the like. In this example, hand model 290 may be referenced by lower arm model 280; lower arm model 280 and upper arm model 270 are referenced by arm model 250; and arm model 250 and leg model 260 are referenced by body model 230, and the like.

[0043] In embodiments of the present invention, not all objects within the object hierarchy need not be referenced. For example, hand model 290 need not reference finger models, but may physically incorporate descriptions of the finger objects within hand model 290. As another example, arm model 250 may include a refernce to hand model 290, but may integraly include the models for the upper arm and the lower arm.

[0044] The heirarchical nature for building-up object models is useful because different users (modellers) are typically assigned the tasks of creating the different models. For example, one modeller is assigned the task of creating hand model 290, a different modeller is assigned the task of creating lower arm model 280, and the like. Accordingly, by dividing-up the responsibility for object creation, the object creation process time is greatly reduced.

[0045] In the present embodiment, animation environment 200 is an environment that allows users (animators) to manipulate object articulation models, via the animation variables (Avars). In one embodiment, animation environment 200 is a Pixar proprietary animation enviroment known as "MenV," although in other embodiments, other animation environments could also be adapted. In this embodiment, animation environment 200 allows an animator to manipulate the Avars provided in the object models (generic rigs) and to move the objects with resspect to time, i.e. animate an object.

[0046] In other embodiments of the present invention, animation environment 200 and object cration environment 210 may be combined into a single integrated environment.

[0047] In Fig. 2, storage system 220 may include any organized and repeatable way to access object articulation models. For example, in one embodiment, storage system 220

includes a simple flat-directory structure on local drive or network drive; in other embodiments, storage system 220 may be an asset management system or a database access system tied to a database, or the like. In the present embodiment, storage system 220 receives references to object models from animation environment 200 and object creation environment 210. In return, storage system 220 provides the object model stored therein. The object model will typically be the "latest" version of the object model that is "checked-into" storage system 220.

[0048] In embodiments of the present invention, as will be discussed further below, object models that include "child" object models, that are incorporated by reference. More particularly, a copy of the referenced object is opened by animation environment 200 or object creation environment 210, but the referenced objects are not physically stored with the "parent" object. For example, lower arm model 280 does not include a copy of hand model 290, but merely a reference to hand model 290. Additionally, lower arm model may include a set of values or parameters that override public attributes for hand model 290.

[0049] Fig. 3 illustrates a block diagram of a flow chart according to embodiments of the present invention. More specifically, Fig. 3 illustrates a process for defining a reusable object model.

[0050] Initially, a user creates an object model and specifies one or more object rigs, step 300. In this embodiment, this can be done by a user using object creation environment 210. For example, a user may specify a series of geometric objects, specify how they are connected, specify control points, specify how they articulated in response to the control points, and the like. In the example of a hand object, the modeler may specify the number of fingers, animation variables, how the fingers articulate in response to the animation variables, and the like. In another example of a generic character rig, the user may reference one or more object rigs, such as a body rig, a head rig, and the like.

[0051] Next, the user determines which attributes of the object model remain private to the object model, step 310. For instance, the modeler determines which properties of the hand object are locked and cannot be modified by users of the hand object. For example, in the case of a hand, the modeler may specify that the number of fingers on the hand are limited to four; the modeler may specify the size of a palm is fixed, or the like. In another instance, the modeler may specify that certain public attributes of referenced object rigs remain public, and certain public attributes are no longer public (i.e. are private).

[0052] In embodiments of the present invention, allowing a user / modeler to specify some model data as private data is valuable. For example, it preserves the right of the modeler to make subsequent changes to an object model that will not conflict with uses of the object model. For instance, if a hand modeler does not lock the number of fingers, subsequent models that refer to the hand model may specify that a hand has six fingers. However at a later time, if the hand modeler decides that hands should have only five fingers, the updated hand model would conflict with the hand object model having six fingers. Accordingly, by keeping certain data private, the hand modeler can make a change from four fingers to five fingers, without causing conflicts in other models. As other examples, the modeler may specify additional deformers that allow the hand to bend in additional ways; the modeler may specify additional deformers that defines new articulation relationships between the fingers; the modeler may specify additional animation variables (avars); and the like. In the present embodiment, as the referring objects request the updated reference model, the additional functionality of the referenced models is automatically provided to the modeler.

[0053] In the present embodiment, the user also determines which attributes of the object model are public for the object model, step 320. The public attributes are attributes that may be specified by other users that reference the object model. For instance, continuing the hand model example, a user / modeler that creates lower arm model 280 may include a reference to hand model 290. In this embodiment, hand model 290 may have a number of public attributes, including length of fingernails, lengths for the different parts of the fingers, length of palm, and the like. In the example of a generic character rig, the user may specify that public attributes of object rigs that are referenced remain public to the user. These attributes may be modified from a default or initial value and overridden by a value specified by the user, as will be illustrated below.

[0054] After the object model is specified, the user stores the object model, step 330. In this embodiment, the object model is provided to storage system 220. For example, the user may store one or more files that specify the object model in a pre-determined location within a particular file path. Alternatively, the user may check-in the object model into an asset management system or a database management system. In such cases, the user would provide one or more files that specified the object model, and one or more keywords, or the like that describe the object. In the example above, the user could provide the hand model file generated by object creation environment 210, and describe the file as a "hand rig," "hand model," or the like.

[0055] Figs. 4A-B illustrate another block diagram of a flow chart according to an embodiment of the present invention. More specifically, Figs. 4A-B illustrate a process for referencing and using a reusable object model.

[0056] In this embodiment, initially, a user creates a new object model or opens an existing object model, step 400. This step typically occurs within object creation environment 210, or the like. Next, the user wishes to retrieve an existing object model, thus the user provides the request to storage system 220, step 410. In one embodiment where storage system 220 is a hierarchical flat-file directory structure, the user may specify or navigate down one or more file paths to find the file name associated with the existing object model. In another embodiment where storage system 220 is an asset management system or a database management system, the user may provide one or more key words for initiating a query. In response, to the user's request, storage system 220 retrieves the requested object model within object creation environment 210, step 420.

[0057] Next, within object creation environment 210, the user can optionally view the "public" attributes of the existing object model, step 430. More particularly, object creation environment 210 provides the user with the limited ability to customize the referred object model by "overriding" default values. In the present embodiment, the user becomes aware of the public attributes, when the user attempts to modify the existing object model. For example, in the example of an existing hand object, when the user modifies the length of a pointer finger, the width of the palm, and the like, the user is modifying the public attributes. In this case, the "private" attributes of the existing object model, cannot be modified by this user, as was previously described. In embodiments of the present invention, the "private" attributes may or may not be displayed to the user at the same time.

[0058] In one embodiment, the user may graphically override values for the public attributes by clicking on one or more control points on the existing object model and moving them, step 440. For example, the user may drag a control point to lengthen a finger of a hand model. In another embodiment, the user may directly specify numeric values, use sliders, or the like to provide values for the public attributes. For example, the user may type-in a value for a thumb width, or the like.

[0059] Next, in this embodiment, the values for the public attributes and a reference to the existing object model are stored in one or more files associated with the new object model, step 450. More specifically, as discussed above, the existing object model exposes public attributes that the user may define values for. In this case, the values entered in step 440 are

saved along with the new object model. For instance, if the new object model is a lower arm model and the referenced model is a hand model, the new object model may include values for the hand model attributes that are changed by the lower arm modeler.

[0060] In other embodiments of the present invention, the user does not specify values for public attributes when creating object models. In such cases, the public attributes of the referenced model become public attributes of the created object.

[0061] Additionally, in this embodiment, the new object model includes a reference to the existing object model that was opened in step 420, above. In one example, the reference includes a directory path in network directory. In other examples, the reference includes one or more key words or reference numbers used by an asset management system or database management system to refer to the existing object model.

[0062] In this example, the user may also perform the steps described in Fig. 3, above, for the new object model. In this way, the user could also define private attributes and public attributes for the new object model. Using the example above of a lower arm model, the user may specify the length of the forearm and the diameter of the forearm as public attributes. The lower arm model can then be referenced by the arm model.

[0063] Additionally, in this embodiment, the user may declare that all attributes of referenced objects that are "public" are now "private." For example, in a lower forearm model that includes a reference to a hand model, all attributes of the hand model can be changed to private. Accordingly, users that reference the lower arm model may not change the geometry of the hand model, or the like. In other embodiments, public attributes of any referenced object, or sub-object can remain public.

[0064] An example of an object model that references another object model is illustrated below. In this example, a first object model stored in the file name "GenericCharacterRig.gpto" was created in Pixar's Geppeto environment. Then, a second object model was created with the file name "Bob.gpto" in Pixar's Geppeto environment. While defining the second object model, the modeler / animator referenced the first object model, and defined values for the public attributes. Then, the second object model was saved. A portion of untitled.gpto is as follows:

[0065] `bob.gpto`

[0066] `#geppetto magic v3.6 $RCSfile$ $Revision$`

[0067] `model "bob" {`

[0068] networkVersion = 1;

[0069] nextId = 28472;

[0070] boneRoot = 4;

[0071] geomRoot = 3;

[0072] deformRoot = 7;

[0073] modelReference "GenericCharacterRig(105)" {

[0074] modelName = GenericCharacterRig;

[0075] modelFile = GenericCharacterRig/GenericCharacterRig.gpto;

[0076] rootProxies = 106 107 108 ;

[0077] deformerContainerProxy "GenericCharacterRig(108)" {

[0078] publicPath = /GenericCharacterRig;

[0079] }

[0080] geomContainerProxy " GenericCharacterRig (107)" {

[0081] publicPath = / GenericCharacterRig;

[0082] }

[0083] boneProxy "CharacterRoot(106)" {

[0084] publicPath = /CharacterRoot;

[0085] }

[0086] geomObjectProxy "Joints_line(28153)" {

[0087] publicPath = /LHandArmature/IndexArmature/Joints_line;

[0088] global_2.point = 16.3526 -7.67344 0.457567;

[0089] }

[0090] }

[0091] ...

[0092] As can be seen in the above example, the first object model is referenced by the following code: "modelReference " GenericCharacterRig (105)" ... modelFile = GenericCharacterRig/GenericCharacterRig.gpto" Additionally, public attributes of

GenericCharacterRig.gpto are specified and set by the following code:" publicPath = LHandArmature/IndexArmature/Joints_line; global_2.point = 16.3526 -7.67344 0.457567."

[0093] In operation, when the animation environment or the object creation environment opens the Bob.gpto file, the latest version of GenericCharacterRig.gpto file is retrieved and the public attributes set in Bob.gpto for GenericCharacterRig.gpto are applied. From the user's perspective, the user is operating upon GenericCharacterRig.gpto, but in reality, the user is merely setting attributes for GenericCharacterRig.gpto within the environment to define the "Bob" character. That is, the local instance of the public attributes are changed by Bob.gpto, but GenericCharacterRig.gpto remains unchanged.

[0094] In the present embodiment, each time a model is referenced, the object creation environment retrieves the latest copy of the referenced model stored in the storage system. Thus, for example, in the situation above, when the user initially defines a lower arm model, a hand model specifies four fingers. Later, the hand model is updated to include five fingers. Subsequently, when the user re-opens the lower arm model, the object creation environment references the latest hand model that specifies five fingers. Because a copy of the referenced object not stored in the new object model, in embodiments, the environment requests the latest "released" or "checked-in" object models. In this embodiment, the referencing functionality is provided to the object creation environment so that it opens the correct reference model, and applies the appropriate set attributes.

[0095] In one embodiment, the user may be automatically notified as to changes in all attributes to referenced models, however in other embodiments, the user may be automatically notified about additions to public attributes, about additions of attributes specified by the referenced object modeler, and even in some embodiments additions to private attributes. As discussed above, if performed correctly, the modeler who defines the referenced model and defines which attributes are public or private, can change private attributes without "breaking" object models that reference the model.

[0096] Fig. 5 illustrates a block diagram of a flow process according to an embodiment of the present invention. More specifically, Fig. 5 illustrates a process of a user (animator) using hierarchical object models, as described herein.

[0097] Initially, a user (animator) invokes an animation environment, step 500. In the present embodiment, the animation environment is Pixar's MenV environment. In this embodiment, the animation environment allows the user (animator) to modify animation

variables (avars) of object models. Based upon the object model, object rigging, and the like, the object model is manipulated in response to the avars.

[0098] Next, with the animation environment, the user requests an object model, step 510. In embodiments of the present invention, the object model is requested from a particular directory location, from an asset management system, a database management system, or the like. As an example, the model "bob.gpto" (example above) is requested. In response, the object model is provided and opened in the animation environment, step 520.

[0099] In the present embodiment, when the model includes a reference to a referenced model, animation environment requests the referenced model, step 530, and the referenced model is opened in the animation environment, step 540. As discussed above, the referenced model returned from the storage system will typically be the latest version of the referenced model. As an example, for the model "bob.gpto" (example above), GenericCharacterRig.gpto is referenced and retrieved. In some embodiments, particular versions of referenced models may be specified an retrieved. For example, a model may specify version 1.0 of a referenced object, or version 3.2 or the like.

[0100] Next, animation environment uses parameters stored in the model and applies them to overwrite the public attributes of the referenced model, step 550. In the present embodiment, a similar process repeats for other referenced objects along the object hierarchy, until all the sub-referenced models are retrieved. For example, in Fig. 2, if the models were all referenced, retrieving body model 240 would include retrieving upper arm model 270, lower arm model 280, hand model 290, and the like, and overriding the respective attributes.

[0101] In some embodiments of the present invention, a model is "flattened out" as an optimization. For example, once a model is "finalized," objects that are referenced are pulled into the final model. Accordingly, the final object model will include the object models of referenced objects. Later, when a user such as an animator overrides the public attributes to create an instance of the object ("bob.gpto" is an instance of GenericCharacterRig.gpto), the instance of the object includes the actual object model along with the overridden attributes. In another embodiment, the instance includes the reference to the "flattened" model and the overridden attributes.

[0102] In other embodiments, the user may specify how "far-down" the hierarchy the user wants to load into the object creation environment. For example, for sake of simplicity, the user may specify that only a limited number sub-referenced object models may be loaded. In

another example, the user may specify only certain branches of the object hierarchy to load. For example, the user (e.g. modeler) may be interested in animating the facial characteristics of a character, accordingly, a leg object referencing an upper thigh object, a lower leg rig, and a foot object, are not loaded. Many other ways to limit the amount of sub-referencing are contemplated in other embodiments.

[0103] Many changes or modifications are readily envisioned. In light of the above disclosure, one of ordinary skill in the art would recognize that the concept of referenced objects may be combined and type of storage systems including network drives, relational databases, check-out system, and the like. Further, in other embodiments, users such as animators, modelers, and the like, may define values for public attributes. Further, once such values are defined, the public attributes may then be declared private, thus the attribute cannot be overridden when referenced. In other embodiments, once values are defined for public attributes, they remain public, and thus overridable when referenced. Other combinations of embodiments are also contemplated.

[0104] In embodiments of the present invention, it is contemplated that multiple "versions" of referenced objects may be supported. For example, a user may create an object and reference a first version of a hand object. Later a second and a third version of the hand object are created. Then when the user reopens the object, the user is presented with the choice of referencing the first version, the second version, or the third version of the hand object. In other embodiments, the user may compare or be given a list of the changes between the first version and the third version. One the user is satisfied with the third version, the object references the third version. Such embodiments are more suitable for asset management systems and/or database embodiments.

[0105] In embodiments where a directory path is used to retrieve object references, the latest version of an object is typically located in the specified location. For example if originally a first version of an object is a filed named hand.gpto, the third version of the object will still be a file named hand.gpto. The older versions of the object may be retrievable from the same directory path, but with different names. In such embodiments therefore, the referenced object will always be the latest version,

[0106] Further embodiments can be envisioned to one of ordinary skill in the art after reading this disclosure. In other embodiments, combinations or sub-combinations of the above disclosed invention can be advantageously made. The block diagrams of the architecture and flow charts are grouped for ease of understanding. However it should be

understood that combinations of blocks, additions of new blocks, re-arrangement of blocks, and the like are contemplated in alternative embodiments of the present invention.

[0107] The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense. It will, however, be evident that various modifications and changes may be made thereunto without departing from the broader spirit and scope of the invention as set forth in the claims.